Polo Aquático Português

João Carlos Machado Rocha Pires (up201806079@fe.up.pt)
Luís Rafael Fernandes Mendes Afonso (up201406189@icbas.up.pt)
Luísa Fernandes Lameiro (up200406067@fe.up.pt)

Bases de Dados, 2019/20 - Turma 7, Grupo 701
Mestrado Integrado em Engenharia Informática e Computação
Faculdade de Engenharia da Universidade do Porto

Descrição

Pretende-se armazenar dados relativos ao Polo Aquático português.

Para cada época, há um conjunto de clubes inscritos na federação. Esses clubes, identificados pelo nome, ano de fundação, região (Norte, Centro, Sul ou Ilhas) e número de títulos, poderão inscrever uma ou mais equipas. Para cada uma dessas equipas, deverá ser indicado o escalão para a qual está inscrita, bem como a listagem dos atletas que a constituem, a classificação obtida na época anterior e a classificação atual.

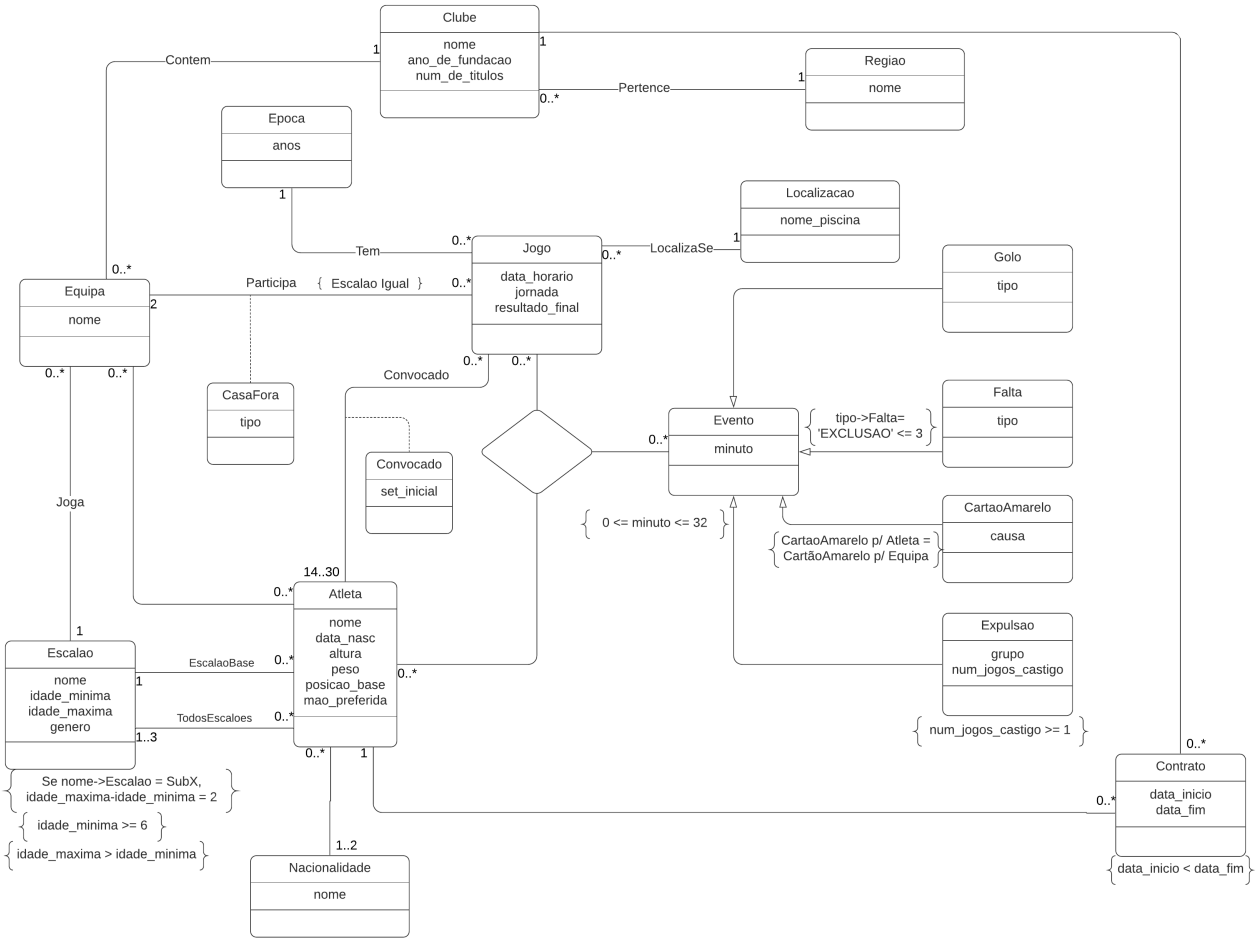
No que ao escalão diz respeito, este deverá conter o nome pelo qual é conhecido, a informação do intervalo de idades permitidas e se se trata de um escalão masculino ou feminino. Quanto à classificação, deverá ser armazenado o total de pontos, a posição na tabela classificativa, os golos marcados e sofridos e o número de jogos efetuados.

Para cada atleta, deverá ser armazenado o nome, a data de nascimento, a nacionalidade, a altura e o peso, a posição em que joga, a mão preferida (esquerda, direita ou ambidestro) e o histórico dos clubes em que já jogou. Este histórico deverá conter, para cada clube, o nome do mesmo e as datas de início e fim.

Para cada época e para cada escalão, existe um conjunto de jogos. Cada jogo tem uma data, indicação da jornada respetiva, local, horário e equipas participantes, bem como o resultado final.

Deverão ainda ser armazenados dados relativos à expulsão de um jogador, se tal ocorrer, identificando o minuto, a causa e o jogador expulso, assim como o número de jogos de castigo atribuídos.

Diagrama UML Revisto



Esquema Relacional

Escalao(idEscalao, nome, idade_minima, idade_maxima, genero)

Atleta(idAtleta, nome, data_nasc, altura, peso, posicao_base, mao_preferida, idEscalao → Escalao)

Nacionalidade(idNacionalidade, nome)

Equipa(idEquipa, nome, idClube → Clube, idEscalao → Escalao)

TodosEscaloes(idEscalao → Escalao, idAtleta → Atleta)

AtletaEquipa(idEscalao → Escalao, idAtleta → Atleta)

NacionalidadeAtleta(idNacionalidade → Nacionalidade, idAtleta → Atetla)

CasaFora(idEquipa → Equipa, idJogo → Jogo, tipo)

Jogo(idJogo, data_horario, jornada, resultado_final, idEpoca → Epoca, idLocalizacao → Localizacao)

Convocado(idAtleta → Atleta, idJogo → Jogo, sete_inicial)

Epoca(idEpoca, anos)

Clube(idClube, nome, ano_de_fundacao, num_de_titulos, idRegiao → Regiao)

Regiao(idRegiao, nome)

Localizacao (idLocalizacao, nome_piscina)

Golo (idEvento → Evento, tipo)

Falta (idEvento → Evento, tipo)

CartaoAmarelo(idEvento → Evento, causa)

Evento(idEvento, minuto)

Expulsao(idEvento → Evento, grupo, num_jogos_castigo)

Contrato(idContrato, data_inicio, data_fim, idAtleta → Atleta, idClube → Clube)

AtletaJogoEvento(idAtleta → Atleta, idEvento → Evento, idJogo → Jogo)

Análise Dependências Funcionais e Formas Normais

Não existem violações, quer à *Boyce-Codd Normal Form*, quer à *3NF*, uma vez que, para todas as dependências funcionais identificadas, os atributos do lado esquerdo são ou a chave primária da relação, ou uma chave alternativa para essa relação.

idEscalao → nome, idade_minima, idade_maxima, genero

idAtleta → nome, data_nasc, altura, peso, posicao_base, mao_preferida, idEscalao

idNacionalidade → nome

idEquipa → nome, idClube, idEscalao

idEquipa → idJogo, tipo

idJogo → data_horario, jornada, resultado_final, idEpoca, idLocalizacao

idAtleta, idJogo → sete_inicial

idEpoca → anos

idClube → nome, ano_de_fundacao, num_de_titulos, idRegiao

idRegiao → nome

idLocalizacao → nome_piscina

idEvento → tipo

idEvento → causa

idEvento → minuto

idEvento → grupo, num_jogos_castigo

idContrato → data_inicio, data_fim, idAtleta, idClube

Lista e Forma de Implementação das Restrições

Ao contrário da Entrega 1, nesta segunda parte do trabalho, por recomendação da Professora, decidimos acrescentar mais restrições, quer ao Diagrama UML, quer ao ficheiro ***criar.sql***. Algumas restrições foram apenas adicionadas ao UML, outras apenas ao ***criar.sql*** e outras ainda aos dois, em simultâneo. De seguida listar-se-ão todas as restrições utilizadas, organizadas pelas classes onde as mesmas foram colocadas.

- Classe Escalao

  Nesta classe, incluímos três restrições. Uma delas foi apenas adicionada ao UML. Restringe o intervalo de idades, ou seja, a diferença entre a idade máxima e a idade mínima do escalão, para 2 se se tratar de um escalão da formação, cujo nome será no formato 'SubX', onde X é um número entre 14, 16 e 18.

  $$\texttt{Se nome} \rightarrow \texttt{Escalao = SubX, idade\_maxima-idade\_minina = 2}$$

  As outras duas restrições foram adicionadas tanto no UML como no ***criar.sql***. No entanto, uma delas sofreu uma ligeira alteração.

  $$\texttt{idade\_minima >= 16}$$

  $$\texttt{idade\_maxima > idade\_minima}$$

  Como se pode verificar no seguinte código, a idade mínima passou a ser 16 dado que o ficheiro ***povoar.sql*** apenas insere nesta tabela dois escalões, Sub20 e Seniores, para os quais a idade mínima é 16 (salvo excepções).

```
create table Escalao    (id INTEGER PRIMARY KEY,
                         nome VARCHAR(8),
                         idade_minima NUMBER,
                         idade_maxima NUMBER,
                         genero CHAR(1),
                         UNIQUE(nome, genero),
                         CHECK(idade_maxima>idade_minima
                         and idade_minima>=16));
```

- Associação Participa

  A restrição nesta associação foi apenas colocada no Diagrama UML. Indica que as duas equipas que se defrontam num jogo terão de pertencer ao mesmo escalão.

  ```
  Escalao Igual
  ```

- Classe Falta

  Nesta classe, a restrição foi também adicionada apenas no Diagrama UML. Indica que o número de faltas do tipo Exclusão apenas pode acontecer, no máximo, 3 vezes para cada atleta. Ao fim das 3 faltas do tipo Exclusão, o atleta fica impossibilitado de continuar a jogar naquele jogo, podendo contudo jogar no jogo imediatamente a seguir, diferenciando-se assim de uma Expulsão.

  $$\texttt{tipo} \rightarrow \texttt{Falta = 'EXCLUSAO' <= 3}$$

- Classe CartaoAmarelo

  Tal como nos dois casos anteriores, foi adicionada apenas no UML uma restrição que indica que, se um jogador for sancionado com um Cartão Amarelo, esse mesmo cartão aplica-se a toda a equipa. Tal significa que um próximo cartão do mesmo tipo, ainda que possa ser dado a outro jogador da mesma equipa, implica duplo Amarelo, ou seja Vermelho. Dito de outra forma, usando um exemplo, imaginemos que o jogador nº 7 vê um Cartão Amarelo. De seguida, o jogador nº 3 comete uma falta também para Cartão Amarelo. O jogador com o nº 3 leva Cartão Vermelho dado que é o seu segundo amarelo (o primeiro Cartão Amarelo não lhe foi mostrado, mas foi a um colega da sua equipa).

  ```
  CartaoAmarelo p/ Atleta = CartaoAmarelo p/ Equipa
  ```

- Classe Evento

  Na classe Evento, adicionamos uma restrição, tanto no UML como no ***criar.sql***, que indica que o minuto pode tomar valores entre o intervalo de 0 a 32, ambos inclusive. Isto porque um jogo de Polo Aquático se divide em 4 períodos, cada um com 8 minutos de tempo útil.

```
create table Evento     (id INTEGER PRIMARY KEY,
                         minuto NUMBER,
                         CHECK(minuto>=0 and minuto<=32));
```

$$0<=minuto<=32$$

- Classe Expulsao

  Na classe Expulsao, adicionamos uma restrição para que o número de jogos de castigo atribuídos seja sempre igual ou superior a 1. Qualquer expulsão implica pelo menos o cumprimento de um jogo de suspensão, podendo ser adicionado um número de jogos de suspensão extra em função do grupo no qual a expulsão se encontra.

```
create table Expulsao(evento INTEGER REFERENCES Evento ON UPDATE CASCADE ON
    DELETE SET NULL,
                      grupo CHAR(2),
                      num_jogos_castigo NUMBER DEFAULT 1,
                      PRIMARY KEY(evento),
                      CHECK(num_jogos_castigo>=1));
```

$$num\_jogos\_castigo >= 1$$

- Classe Contrato

  Para a classe Contrato, adicionamos uma restrição que garante que a sua data de fim é posterior à sua data de início.

```
create table Contrato(id INTEGER PRIMARY KEY,
                      data_inicio TEXT,
                      data_fim TEXT,
                      atleta INTEGER REFERENCES Atleta ON UPDATE CASCADE ON
   DELETE SET NULL,
                      clube INTEGER REFERENCES Clube ON UPDATE CASCADE ON
   DELETE SET NULL,
                      CHECK(data_inicio<data_fim));
```

```
data_inicio < data_fim
```

- Classe Epoca

  Na Classe Epoca, para garantir que o ano de início era um valor, no mínimo, realista, adicionamos uma restrição apenas no ***criar.sql*** para garantir que o mesmo era superior a 1900.

```
create table Epoca      (id INTEGER PRIMARY KEY,
                         anoInicio NUMBER UNIQUE NOT NULL
                         CHECK(anoInicio>=1900));
```

```
anoInicio >= 1900
```

- Classe Jogo

  Na Classe Jogo, adicionamos uma restrição que obriga a que a Jornada não só seja positiva como igual ou superior a 1.

```
create table Jogo   (id INTEGER PRIMARY KEY,
                     data_horario TEXT,
                     jornada NUMBER CHECK(jornada>=1),
                     resultado_final VARCHAR(5) DEFAULT '0-0',
                     epoca INTEGER REFERENCES Epoca ON UPDATE CASCADE ON
    DELETE SET NULL,
                     localizacao INTEGER REFERENCES Localizacao ON UPDATE
    CASCADE ON DELETE SET NULL);
```

```
jornada >= 1
```

- Classe Clube

  Na Classe Clube, adicionamos uma restrição dupla, que obriga a que o ano de fundação de um Clube seja igual ou superior a 1900, tal como o ano de início para uma Época (definido na Classe Epoca), e que o número de títulos seja igual ou superior a 0.

```
create table Clube  (id INTEGER PRIMARY KEY,
                     nome TEXT UNIQUE,
                     ano_de_fundacao NUMBER,
                     num_de_titulos NUMBER DEFAULT 0,
                     id_regiao INTEGER REFERENCES Regiao ON UPDATE CASCADE
    ON DELETE SET NULL,
                     CHECK(ano_de_fundacao>=1900 and num_de_titulos>=0));
```

$$ano\_de\_fundacao >= 1900$$

$$num\_de\_titulos >= 0$$

- Classe Atleta

  Finalmente, para a Classe Atleta, adicionamos duas restrições, para o peso e para a altura de cada jogador, não permitindo que sejam inferiores a 30 e a 150, respectivamente.

```
create table Atleta     (id INTEGER PRIMARY KEY,
                         nome TEXT,
                         data_nasc TEXT,
                         altura NUMBER,
                         peso NUMBER,
                         posicao_base TEXT,
                         mao_preferida CHAR(1) DEFAULT 'A',
                         escalao INTEGER REFERENCES Escalao ON UPDATE
   CASCADE ON DELETE SET NULL,
                         CHECK(peso>=30 and altura>=150));
```

$$peso >= 30$$

$$altura >= 150$$